NAG Library Function Document

nag_quad_1d_gauss_wgen (d01tcc)

1 Purpose

nag_quad_1d_gauss_wgen (d01tcc) returns the weights (normal or adjusted) and abscissae for a Gaussian integration rule with a specified number of abscissae. Six different types of Gauss rule are allowed.

2 Specification

```
#include <nag.h>
#include <nagd01.h>
```

```
void nag_quad_1d_gauss_wgen (Nag_QuadType quad_type, double a, double b,
     double c, double d, Integer n, double weight[], double abscis[],
     NagError *fail)
```

3 Description

nag_quad_1d_gauss_wgen (d01tcc) returns the weights w_i and abscissae x_i for use in the summation

$$S = \sum_{i=1}^{n} w_i f(x_i)$$

which approximates a definite integral (see Davis and Rabinowitz (1975) or Stroud and Secrest (1966)). The following types are provided:

(a) Gauss–Legendre

$$S \simeq \int_a^b f(x)\,dx, \qquad \text{exact for } f(x) = P_{2n-1}(x).$$

Constraint: $b > a$.

(b) Gauss–Jacobi

normal weights:

$$S \simeq \int_a^b (b-x)^c (x-a)^d f(x)\,dx, \qquad \text{exact for } f(x) = P_{2n-1}(x),$$

adjusted weights:

$$S \simeq \int_a^b f(x)\,dx, \qquad \text{exact for } f(x) = (b-x)^c (x-a)^d P_{2n-1}(x).$$

Constraint: $c > -1,\ d > -1,\ b > a$.

(c) Exponential Gauss

normal weights:

$$S \simeq \int_a^b \left| x - \frac{a+b}{2} \right|^c f(x)\,dx, \qquad \text{exact for } f(x) = P_{2n-1}(x),$$

adjusted weights:

$$S \simeq \int_a^b f(x)\,dx, \qquad \text{exact for } f(x) = \left|x - \frac{a+b}{2}\right|^c P_{2n-1}(x).$$

Constraint: $c > -1,\ b > a$.

(d) Gauss–Laguerre

normal weights:

$$\begin{aligned} S &\simeq \int_a^\infty |x-a|^c e^{-bx} f(x)\,dx \quad (b > 0), \\ &\simeq \int_{-\infty}^a |x-a|^c e^{-bx} f(x)\,dx \quad (b < 0), \qquad \text{exact for } f(x) = P_{2n-1}(x), \end{aligned}$$

adjusted weights:

$$\begin{aligned} S &\simeq \int_a^\infty f(x)\,dx \quad (b > 0), \\ &\simeq \int_{-\infty}^a f(x)\,dx \quad (b < 0), \qquad \text{exact for } f(x) = |x-a|^c e^{-bx} P_{2n-1}(x). \end{aligned}$$

Constraint: $c > -1,\ b \neq 0$.

(e) Gauss–Hermite

normal weights:

$$S \simeq \int_{-\infty}^{+\infty} |x-a|^c e^{-b(x-a)^2} f(x)\,dx, \qquad \text{exact for } f(x) = P_{2n-1}(x),$$

adjusted weights:

$$S \simeq \int_{-\infty}^{+\infty} f(x)\,dx, \qquad \text{exact for } f(x) = |x-a|^c e^{-b(x-a)^2} P_{2n-1}(x).$$

Constraint: $c > -1,\ b > 0$.

(f) Rational Gauss

normal weights:

$$\begin{aligned} S &\simeq \int_a^\infty \tfrac{|x-a|^c}{|x+b|^d} f(x)\,dx \quad (a+b > 0), \\ &\simeq \int_{-\infty}^a \tfrac{|x-a|^c}{|x+b|^d} f(x)\,dx \quad (a+b < 0), \qquad \text{exact for } f(x) = P_{2n-1}\left(\tfrac{1}{x+b}\right), \end{aligned}$$

adjusted weights:

$$\begin{aligned} S &\simeq \int_a^\infty f(x)\,dx \quad (a+b > 0), \\ &\simeq \int_{-\infty}^a f(x)\,dx \quad (a+b < 0), \qquad \text{exact for } f(x) = \tfrac{|x-a|^c}{|x+b|^d} P_{2n-1}\left(\tfrac{1}{x+b}\right). \end{aligned}$$

Constraint: $c > -1,\ d > c+1,\ a+b \neq 0$.

In the above formulae, $P_{2n-1}(x)$ stands for any polynomial of degree $2n-1$ or less in x.

The method used to calculate the abscissae involves finding the eigenvalues of the appropriate tridiagonal matrix (see Golub and Welsch (1969)). The weights are then determined by the formula

$$w_i = \left\{ \sum_{j=0}^{n-1} P_j^*(x_i)^2 \right\}^{-1}$$

where $P_j^*(x)$ is the jth orthogonal polynomial with respect to the weight function over the appropriate interval.

The weights and abscissae produced by nag_quad_1d_gauss_wgen (d01tcc) may be passed to nag_quad_md_gauss (d01fbc), which will evaluate the summations in one or more dimensions.

4 References

Davis P J and Rabinowitz P (1975) *Methods of Numerical Integration* Academic Press

Golub G H and Welsch J H (1969) Calculation of Gauss quadrature rules *Math. Comput.* **23** 221–230

Stroud A H and Secrest D (1966) *Gaussian Quadrature Formulas* Prentice–Hall

5 Arguments

1: **quad_type** – Nag_QuadType *Input*

On entry: indicates the type of quadrature rule.

quad_type = Nag_Quad_Gauss_Legendre
Gauss–Legendre, with normal weights.

quad_type = Nag_Quad_Gauss_Jacobi
Gauss–Jacobi, with normal weights.

quad_type = Nag_Quad_Gauss_Jacobi_Adjusted
Gauss–Jacobi, with adjusted weights.

quad_type = Nag_Quad_Gauss_Exponential
Exponential Gauss, with normal weights.

quad_type = Nag_Quad_Gauss_Exponential_Adjusted
Exponential Gauss, with adjusted weights.

quad_type = Nag_Quad_Gauss_Laguerre
Gauss–Laguerre, with normal weights.

quad_type = Nag_Quad_Gauss_Laguerre_Adjusted
Gauss–Laguerre, with adjusted weights.

quad_type = Nag_Quad_Gauss_Hermite
Gauss–Hermite, with normal weights.

quad_type = Nag_Quad_Gauss_Hermite_Adjusted
Gauss–Hermite, with adjusted weights.

quad_type = Nag_Quad_Gauss_Rational
Rational Gauss, with normal weights.

quad_type = Nag_Quad_Gauss_Rational_Adjusted
Rational Gauss, with adjusted weights.

Constraint: **quad_type** = Nag_Quad_Gauss_Legendre, Nag_Quad_Gauss_Jacobi, Nag_Quad_Gauss_Jacobi_Adjusted, Nag_Quad_Gauss_Exponential, Nag_Quad_Gauss_Exponential_Adjusted, Nag_Quad_Gauss_Laguerre, Nag_Quad_Gauss_Laguerre_Adjusted, Nag_Quad_Gauss_Hermite, Nag_Quad_Gauss_Hermite_Adjusted, Nag_Quad_Gauss_Rational or Nag_Quad_Gauss_Rational_Adjusted.

2: **a** – double *Input*
3: **b** – double *Input*
4: **c** – double *Input*
5: **d** – double *Input*

On entry: the parameters a, b, c and d which occur in the quadrature formulae. **c** is not used if **quad_type** = Nag_Quad_Gauss_Legendre; **d** is not used unless **quad_type** = Nag_Quad_Gauss_Jacobi, Nag_Quad_Gauss_Jacobi_Adjusted, Nag_Quad_Gauss_Rational or Nag_Quad_Gauss_Rational_Adjusted. For some rules **c** and **d** must not be too large (see Section 6).

Constraints:

if **quad_type** = Nag_Quad_Gauss_Legendre, $\mathbf{a} < \mathbf{b}$;
if **quad_type** = Nag_Quad_Gauss_Jacobi or Nag_Quad_Gauss_Jacobi_Adjusted, $\mathbf{a} < \mathbf{b}$ and $\mathbf{c} > -1.0$ and $\mathbf{d} > -1.0$;
if **quad_type** = Nag_Quad_Gauss_Exponential or Nag_Quad_Gauss_Exponential_Adjusted, $\mathbf{a} < \mathbf{b}$ and $\mathbf{c} > -1.0$;
if **quad_type** = Nag_Quad_Gauss_Laguerre or Nag_Quad_Gauss_Laguerre_Adjusted, $\mathbf{b} \neq 0.0$ and $\mathbf{c} > -1.0$;
if **quad_type** = Nag_Quad_Gauss_Hermite or Nag_Quad_Gauss_Hermite_Adjusted, $\mathbf{b} > 0.0$ and $\mathbf{c} > -1.0$;
if **quad_type** = Nag_Quad_Gauss_Rational or Nag_Quad_Gauss_Rational_Adjusted, $\mathbf{a} + \mathbf{b} \neq 0.0$ and $\mathbf{c} > -1.0$ and $\mathbf{d} > \mathbf{c} + 1.0$.

6: **n** – Integer *Input*

On entry: n, the number of weights and abscissae to be returned. If **quad_type** = Nag_Quad_Gauss_Exponential_Adjusted or Nag_Quad_Gauss_Hermite_Adjusted and $\mathbf{c} \neq 0.0$, an odd value of **n** may raise problems (see **fail.code** = NE_INDETERMINATE).

Constraint: $\mathbf{n} > 0$.

7: **weight**[**n**] – double *Output*

On exit: the **n** weights.

8: **abscis**[**n**] – double *Output*

On exit: the **n** abscissae.

9: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_CONSTRAINT

On entry, **a**, **b**, **c**, or **d** is not in the allowed range: **a** = ⟨*value*⟩, **b** = ⟨*value*⟩ **c** = ⟨*value*⟩, **d** = ⟨*value*⟩ and **quad_type** = ⟨*value*⟩.

NE_CONVERGENCE

The algorithm for computing eigenvalues of a tridiagonal matrix has failed to converge.

NE_INDETERMINATE

Exponential Gauss or Gauss–Hermite adjusted weights with **n** odd and **c** $\neq 0.0$.

Theoretically, in these cases:

for **c** > 0.0, the central adjusted weight is infinite, and the exact function $f(x)$ is zero at the central abscissa;

for **c** < 0.0, the central adjusted weight is zero, and the exact function $f(x)$ is infinite at the central abscissa.

In either case, the contribution of the central abscissa to the summation is indeterminate.

In practice, the central weight may not have overflowed or underflowed, if there is sufficient rounding error in the value of the central abscissa.

The weights and abscissa returned may be usable; you must be particularly careful not to 'round' the central abscissa to its true value without simultaneously 'rounding' the central weight to zero or ∞ as appropriate, or the summation will suffer. It would be preferable to use normal weights, if possible.

Note: remember that, when switching from normal weights to adjusted weights or vice versa, redefinition of $f(x)$ is involved.

NE_INT

On entry, **n** $= \langle$*value*$\rangle$.
Constraint: **n** > 0.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

NE_TOO_BIG

One or more of the weights are larger than $rmax$, the largest floating point number on this computer (see nag_real_largest_number (X02ALC)): $rmax = \langle$*value*$\rangle$.
Possible solutions are to use a smaller value of n; or, if using adjusted weights to change to normal weights.

NE_TOO_SMALL

One or more of the weights are too small to be distinguished from zero on this machine.
The underflowing weights are returned as zero, which may be a usable approximation.
Possible solutions are to use a smaller value of n; or, if using normal weights, to change to adjusted weights.

7 Accuracy

The accuracy depends mainly on n, with increasing loss of accuracy for larger values of n. Typically, one or two decimal digits may be lost from machine accuracy with $n \simeq 20$, and three or four decimal digits may be lost for $n \simeq 100$.

8 Parallelism and Performance

nag_quad_1d_gauss_wgen (d01tcc) is not threaded by NAG in any implementation.

nag_quad_1d_gauss_wgen (d01tcc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The major portion of the time is taken up during the calculation of the eigenvalues of the appropriate tridiagonal matrix, where the time is roughly proportional to n^3.

10 Example

This example returns the abscissae and (adjusted) weights for the seven-point Gauss–Laguerre formula.

10.1 Program Text

```
/* nag_quad_1d_gauss_wgen (d01tcc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagd01.h>

int main(void)
{
  Integer exit_status = 0;
  Integer i, n;
  double  a, b, c, d;
  Nag_QuadType quadtype;
  NagError fail;
  double   *abscis = 0, *weight = 0;

  INIT_FAIL(fail);

  printf("nag_quad_1d_gauss_wgen (d01tcc) Example Program Results\n");
  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  /* Input a, b, c, d and n */
#ifdef _WIN32
  scanf_s("%lf %lf %lf %lf", &a, &b, &c, &d);
#else
  scanf("%lf %lf %lf %lf", &a, &b, &c, &d);
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%*[^\n] ", &n);
#else
  scanf("%"NAG_IFMT"%*[^\n] ", &n);
#endif
  quadtype = Nag_Quad_Gauss_Laguerre_Adjusted;

  if (!(abscis = NAG_ALLOC(n, double)) ||
```

```
      !(weight = NAG_ALLOC(n, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* nag_quad_1d_gauss_wgen (d01tcc).
   * Calculation of weights and abscissae for
   * Gaussian quadrature rules, general choice of rule.
   */
  nag_quad_1d_gauss_wgen(quadtype, a, b, c, d, n, weight, abscis, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_quad_1d_gauss_wgen (d01tcc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

  printf("\nLaguerre formula, %3"NAG_IFMT " points\n\n"
         "    Abscissae        Weights\n\n", n);
  for (i = 0; i < n; i++)
    {
      printf("%15.5e", abscis[i]);
      printf("%15.5e\n", weight[i]);
    }
  printf("\n");

 END:
  NAG_FREE(abscis);
  NAG_FREE(weight);

  return exit_status;
}
```

10.2 Program Data

```
nag_quad_1d_gauss_wgen (d01tcc) Example Program Data
   0.0   1.0   0.0   0.0
   7
```

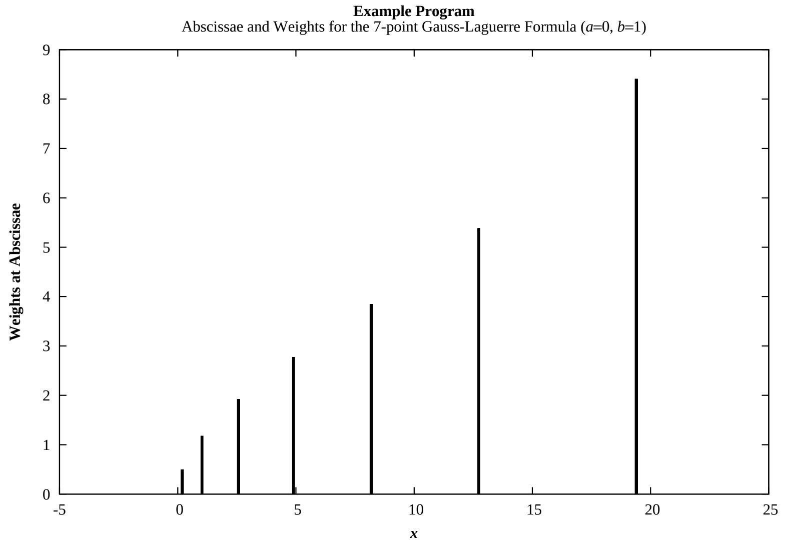
10.3 Program Results

```
nag_quad_1d_gauss_wgen (d01tcc) Example Program Results

Laguerre formula,   7 points

    Abscissae        Weights

    1.93044e-01    4.96478e-01
    1.02666e+00    1.17764e+00
    2.56788e+00    1.91825e+00
    4.90035e+00    2.77185e+00
    8.18215e+00    3.84125e+00
    1.27342e+01    5.38068e+00
    1.93957e+01    8.40543e+00
```


Example Program
Abscissae and Weights for the 7-point Gauss-Laguerre Formula (a=0, b=1)
Weights at Abscissae
x
0
1
2
3
4
5
6
7
8
9
-5
0
5
10
15
20
25